PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-312152
(43)Date of publication of application : 09.11.1999

(51)Int.Cl. G06F 15/16
H04Q 7/38

(21)Application number : 10-365666
(22)Date of filing : 22.12.1998

(71)Applicant : TEXAS INSTR INC <TI>
(72)Inventor : MCMAHON MICHAEL
LINEBERRY MARION C
WOOLSEY MATTHEWS A
CHAUVEL GERARD

(30)Priority
Priority number : 97 995606 Priority date : 22.12.1997 Priority country : US

(54) MOBILE ELECTRONIC DEVICE AND CONTROL METHOD THEREFOR

(57)Abstract:
PROBLEM TO BE SOLVED: To obtain data processing architecture for improving and optimizing the use of many processors and coprocessors.
SOLUTION: A wireless data platform 10 includes plural processors 1216. A communication channel is opened between processorsso as to communicate information at the time of executing a task. A dynamic cross compiler 80 to be executed by a single processor is compiled to a native processing code for the other processor. A dynamic cross linker 82 links the code compiled for the other processor. It is also available to download a native code to the platform by using JAVA Bean (or another language type)including the native code. The JAVA Bean can be ciphered as a digital sign for security.

CLAIMS

[Claim(s)]
[Claim 1]A move electronic device comprising:
A co-processor which executes a native code.
It is a host processor system which operates so that a native code corresponding to a host processor system and a processor independent code may be executedA communication circuit between a host processor system which operates so that a

task carried out by a digital signal co-processor may be changed dynamicallysaid host processor systemand said co-processor.

[Claim 2]A control method of a move electronic device characterized by comprising the following.
A step which executes a native code in a co-processor.
A step which executes a native code and a processor independent code in a host processor system.
A step which changes dynamically a task carried out by a digital signal co-processor by said host processor system.
A step which communicates between said host processor system and said co-processor.

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention]Especially this invention relates to the hardware for move electronic devicesand a software platformconcerning a move electronic device generally.
[0002]
[Description of the Prior Art]The popularity of a handheld computer portable instrument is increasing as the power of a devicetherefore function ** are reinforced. PDA (Personal DigitalAssistant) is used widely in recent yearsand the smart phone (Smart phones) which combined some capability of a cellular phone machine and PDA is considered to give remarkable impact to communication of the near future.
[0003]The general-purpose processor for one or more DSPs (digital signal processor) which provide some individual functionssuch as speech recognitionor other co-processorsand other data processing functions is built in a certain device now. The code for DSP and the code for general-purpose processors are generally stored in ROM and other nonvolatile memoryand they cannot be corrected easily. Thereforewhen an improvement and a new function can be usedcapability of a device may often be unable to be upgraded. DSP or other co-processors which may exist in a device especially cannot be used for the maximum.
[0004]
[Problem(s) to be Solved by the Invention]Thereforethe data-processing architecture which can upgrade and optimize use of many processors and a co-processor is needed.
[0005]
[Means for Solving the Problem]In this inventiona move electronic device contains a co-processor which executes a native codeand a host processor system which

operates so that a native code and a processor independent code corresponding to a host processor system may be executed. The host processor system operates so that a task carried out by a digital signal co-processor may be changed dynamically. Communication between a host processor system and a co-processor is provided by communication circuit.
[0006]A remarkable advantage which endures conventional technology by this invention is provided. It is because a host processor system assigns dynamically a task carried out by a co-processor which may be a digital signal processor and can use a co-processor thoroughly. According to various factorssuch as the present capability of each processorthe routine of the host processor system can be directly carried out to one of two or more of the co-processors.
[0007]
[Embodiment of the Invention]Desirable working example of the general wireless data platform architecture which can be used for carrying out a smart phone and PDA is shown in drawing 1. The wireless data platform 10 contains the wide use (host) processor 12 connected to the bus structure 14 including the data bus 14athe address bus 14band the control bus 14c. One or more DSP(or other co-processors) 16 including the core processor 16a and the peripheral interface 16b The bus 14 and the DSP cache memory 18alt is connected to CPU cache 18bthe memory containing MMU(memory management unit)18cand the traffic controller 18. The Hardware accelerator circuit 20 (portable languagessuch as JAVAare accelerated)videoand LCD controller 22 are also connected to the memory and the traffic controller 18. The output of video and an LCD controller is connected to LCD or the video display 24.
[0008]The memory and the traffic controller 18 are connected to the main memory unit 26 shown as the bus 14 and a SDRAM (synchronous dynamic random access memory). The bus 14 is connected also to I/O controller 28Interface Division 30and RAM/ROM32. Two or more devices are connectable with the one or more serial ports 40such as the smart card 34the keyboard 36the wireless data platform 10 of mouse 38 gradea USB (Universal Serial Bus) portor RS232 serial port. Interface Division 30 is connectable with the flash memory card 42 and/or the DRAM card 44. The peripheral interface 16b can connect DSP16 to DAC(digital/analog converter) 46the network interface 48or other devices.
[0009]The wireless data platform 10 of drawing 1 uses general-purpose processor 12 and DSP16. Unlike the present device carried out only for the fixing function with specific DSP16DSP16 of drawing 1 is applicable to the function of any number. Therebythe user can pull out the profits of DSP16 thoroughly.
[0010]One main area which can use DSP16 relates to the man machine interface (MMI). It is important that functionssuch as speech recognitionan image and video compandingdata encryptionand text voice conversioncan be more efficiently carried out using DSP16. A new function and improvement can be easily added to the wireless data platform 10 by this architecture.
[0011]The wireless data platform 10 is a common block diagramand is variously correctable. For exampleindependent DSP and the processor cash 18a and 18b

are illustrated by drawing 1. Unit type cash can also be used for a person skilled in the art like understanding. The Hardware acceleration (acceleration) circuit 20 is an optional item. Although execution of languagessuch as JAVAis accelerated by such a devicethis circuit is not required for operation of a device. Although only one DSP is illustratedmany DSPs (or other co-processors) are connectable with a bus.

[0012]The functional software architecture of the wireless data platform 10 is shown in drawing 2. This block diagram is assumed using JAVA and languages other than JAVA can also be used. Functionallysoftware is divided to two groups and it is host processor software and DSP software. Host software contains the one or more applets 40. The DSP API class 42 is an applet accessed to the function of the JAVA API package for JAVA applicationsi.e.DSP API50and the host DSP Interface Division layer 52. The JAVA virtual machine (VM) 44 interprets an applet. JAVA native Interface Division 46 is the way JAVA VM executes a host processor or a specific platform code. The native task 48 is a non-JAVA program which can be executed by the host processor 12 without using JAVA native Interface Division. DSP API50 mentioned later is API (application programming interface) used by the host 12 who calls in order to use the capability of DSP16. The host DSP Interface Division layer 52 provides host 12 and DSP16 with API for communicating with other hardwares mutually using the channel through other tasks or host DSP communications protocols. the host RTOS for the DSP device driver 54 to communicate with DSP16 -- it is a host base device driver 56 (real time operating system) business. Host RTOS56 are operating systemssuch as NUCLEUS PLUS of an AKUSERARETEDDO technology company. Non-real time operating systemssuch as WINDOWSCE of Microsoft Corp.can also be used. The program stored in order to perform by DSP16 in the DSP library 58 is included.

[0013]By the DSP sidethe one or more tasks 60 performed by DSP16 are storable in a memory. A task can be taken in and out of a memory by request so that it may mention laterand it may become that the function of DSP is not staticand dynamic. The host DSP Interface Division layer 62 by the side of DSP carries out the same function as the host DSP Interface Division layer 52 by the side of a hostnamelyhost 12 and DSP16 can communicate. DSP RTOS64 is an operating system for DSP processors. The host device driver 66 is a DSP base device driver for DSP RTOS64 who communicates with the host 12. Host DSP Interface Division 70 connects DSP16 to the host 12.

[0014]In operationthe software architecture shown in drawing 2 uses DSP16 not as a fixing function device but as a variable function device like conventional technology. Thereforea DSP function is downloaded to the moving system which contains the architecture of drawing 2and DSP16 can make it possible to carry out various signal processing functions to the host 12.

[0015]DSP-APIDSP-API provides device independent Interface Division from the host 12 to DSP16. The capability to schedule by loading a task to the host 12 DSP16and to control these tasksand to communicate with this function is given. The available resources of DSP are determinedthe host 12 and a DSP task are

generated and controlled in an API functionthe data channel between the host 12 and a DSP task is generated and controlled to itand the call for communicating with a task is contained in it. These functions are mentioned later. Each function returns a Boolean result and it is SUCCESS to the operation in which it succeededor FAILURE. If a result is FAILUREit must check which error errcode was checked and has occurred.

[0016]

[Equation 1]

This function returns the present MIPS which can be used on DSP. It is the sum of MIPS rating to the MIPS capability minus base MIPS value (a MIPS value without an addition dynamic taski.e.the kernel plus API code phillips screwdriver) of DSP16and all the dynamic tasks by which the minus load was carried out. An errcode parameter includes the result the following is considered to be.

[Equation 2]

[0017]

[Equation 3]

This function is asked to DSP16 specified by DevID about the memory quantity which can be used to both program memory and data memory. The value obtained as a result is returned in progmem and a datamem parameter. Size is specified in TDSPWords.An errcode parameter includes the result the following is considered to be.

[Equation 4]

[0018]

[Equation 5]

A 1-block memory is allocated by this function on DSP16. It is specified on which device DevID allocates a memory. mempage is 0 to a program space and is 1 to data space. A size parameter specifies the memory-block size in TDSPWords.memptr returned is a pointer to memory block on DSP16or NULL at the time of failure. An errcode parameter includes the result the following is considered to be.

[Equation 6]

[0019]

[Equation 7]

This function releases the 1-block memory on DSP which was able to assign the DSPAllocMem function.DevID specifies on which device a memory resides permanently. mempage is 0 to a program space and is 1 to data space. A memptr parameter is a pointer to memory block. An errcode parameter includes the result the following is considered to be.
[Equation 8]

[0020]
[Equation 9]

This function is accessed to a DSP library tableand the code header of the DSP function code specified with a Name parameter is returned. When returnedthe position directed with a codehdr parameter includes code header information. An errcode parameter includes the result the following is considered to be.
[Equation 10]

[0021]
[Equation 11]

This function links a DSP function code so that it may run in a directed address on DSP specified by DevID. A codehdr parameter directs a code header of a function. A dynamic crossing linker links a code based on information in a code header and a code (COFF file). A dynamic crossing linker allocates a memory if neededand links and loads a code to DSP16. A tcs parameter is a pointer to a required creation-of-task structure in a DSPCreateTask function.DSPLinkCode is filled up with a code entry pointa priorityand the quantum (quantum) field of structure in preparation for a creation of task.An errcode parameter includes a result the following is considered to be.
[Equation 12]

[0022]
[Equation 13]

This function copies specified BLOB (Binary Large Object) to DSP16. It is specified to which DSP16 DevID copies an object. A srcaddr parameter is a

pointer to the object in a host memory. destaddr is a pointer to the position which copies the object on DSP16. mempage is 0 to a program space and is 1 to data space. A size parameter specifies the size of the object in TDSPWords.An errcode parameter includes the result the following is considered to be.
[Equation 14]

[0023]
[Equation 15]

It is required for DSP16 that DSPCreateTask will generate a task if the code position in a task parameter and the program space of DSP is given.Creation-of-task structure is shown in Table 1.
[0024]
[Table 1]

[0025]If a task is generateda Create entry point will be called and an opportunity to perform arbitrary required preliminary initialization will be given to a task. CreateSuspendResumeand a Stop entry point can be set to NULL. TaskID obtained as a result contains both a device ID (DevID) and task ID of DSP. Generation is failure if TaskID is NULL. An errcode parameter includes the result the following is considered to be.
[Equation 16]

[0026]
[Equation 17]

This function starts the DSP task specified by TaskID. Execution begins by the Start entry point of a task. An errcode parameter includes the result the following is considered to be.
[Equation 18]

[0027]
[Equation 19]

This function suspends the DSP task specified by TaskID. Before being suspendedin order to give an opportunity to carry out arbitrary required housekeeping to a taskthe Suspend entry point of a task is called. An errcode

parameter includes the result the following is considered to be.
[Equation 20]

[0028]
[Equation 21]

This function resumes the DSP task suspended by DSPSuspendTask.Before being resumedin order to give an opportunity to carry out arbitrary required housekeeping to a taskthe Resume entry point of a task is called. An errcode parameter includes the result the following is considered to be.
[Equation 22]

[0029]
[Equation 23]

This function deletes the DSP task specified by TaskID. Before deletingin order to give an opportunity to carry out arbitrary required cleanups to a taskthe Stop entry point of a task is called. It must include that it returns the arbitrary resources which released the arbitrary memories allocated by the task and the task acquired. An errcode parameter includes the result the following is considered to be.
[Equation 24]

[0030]
[Equation 25]

This function changes the DSP priority of a task specified by TaskID. A priority is changed to newpriority. The value newpriority is considered to be is RTOS dependence. If it returnsan oldpriority parameter will be set as the priority in front of a task. An errcode parameter includes the result the following is considered to be.
[Equation 26]

[0031]
[Equation 27]

This function returns a DSP task status specified by TaskID. status contains one of the following values.
[Equation 28]

[0032]A priority parameter contains a priority of a taskand Input and an Output parameter contain eachthe inputand output ID of a task. An errcode parameter includes the result the following is considered to be.
[Equation 29]

[0033]
[Equation 30]

This function returns ID of the object which attached the name on DSP16. The object which named can be made into the DSP object which attached a channela taskmemory blockor the supported name of arbitrary others. An errcode parameter includes the result the following is considered to be.
[Equation 31]

[0034]
[Equation 32]

This function requires 1 block of a memory. mempage specifies program memory (0) or data memory (1). An addr parameter specifies a memory start address and count shows which reads many TSDPWords(es).A buf parameter is a pointer to the call person who provided the buffer which must copy a memory. An errcode parameter includes the result the following is considered to be.
[Equation 33]

[0035]
[Equation 34]

This function writes in 1 block of a memory. mempage specifies program memory (0) or data memory (1). An addr parameter specifies a memory start address and count shows which writes in many TSDPWords(es).A buf parameter is a pointer to the buffer containing the memory to write in. An errcode parameter includes the result the following is considered to be.
[Equation 35]

[0036]
[Equation 36]

This function reads a DSP register and returns the value in regvalue. It is specified which register the RegID parameter should return. If RegID is −1all the register values will be returned. The regvalue parameter which is a pointer to the calling party who provided the buffer must direct sufficient memory storage to hold all the values. Register ID is for specific DSPs and is dependent on a specific implementation. An errcode parameter includes the result the following is considered to be.
[Equation 37]

[0037]
[Equation 38]

This function is written in a DSP register. It is specified which register a RegID parameter corrects. regvalue contains the new value to write in. Register ID is for specific DSPs and is dependent on a specific implementation. An errcode parameter includes the result the following is considered to be.
[Equation 39]

[0038]
[Equation 40]

This function sets up the break point (break point) in a given code address (addr). An errcode parameter includes the result the following is considered to be.
[Equation 41]

[0039]
[Equation 42]

This function clears the break point beforehand set up by DSPDbgSetBreak in the given code address (addr).An errcode parameter includes the result the following is considered to be.
[Equation 43]

[0040]The DSP device driver DSP device driver 54 processes the communication to DSP16 from the host 12. A driver function takes up the needed information specified in the host DSP communications protocoland processes transmission of the information through an available hardware interface. Device drivers are RTOS dependence and communication hardware dependence.
[0041]The DSP library DSP library 58 includes the block of the code which can be downloaded and executed to DSP16. Each block of the code is beforehand linked relocatable as un-linkingi.e.a libraryso that a dynamic crossing linker can opt for all the address reference (reference). Each code block also includes DSP MIPS (million instruction per second)a prioritya time slice quantumand the information about the demand of a block to a memory. The format of a code block header is shown in Table 2. Program memory and data memory size are the approximate values for giving the host 12 the quick check of whether DSP can support the memory request of a task. If it seems that there is sufficient spacethe dynamic crossing linker can try the link and loading of a code. By page alignment (alignment) and demand of continuity (contiguity)the dynamic crossing linker still may be unable to achieve the purpose. In desirable working examplea code is a version 2COFF file format.
[0042]
[Table 2]

[0043]A procedure changed into a target code to which portable (processor independence) codessuch as the conversion JAVA code to a target code to which a portable code was linkedwere linked is shown in drawing 3. This procedure uses two functionsthe dynamic cross compiler 80and the dynamic crossing linker 82. Each function is realized on the host processor 12. In desirable working examplea dynamic crossing linker is a part of DSP-API. A cross compiler can also be made into a part of DSPAPI.
[0044]The dynamic cross compiler 80 is changed into a excecutable target processor code to which a portable code is not linked. The dynamic crossing linker 82 is changed into a excecutable target processor code to which un-linking and a excecutable target processor code were linked. In order to do sobefore carrying out loading to up to DSP16an address in a 1-block code must be determined. The dynamic crossing linker 82 links a code segment and a data segment of a functionallocates a memory on DSP16and loads a code and fixed data to DSP16. Since a function (compiling and linking) arises in a different processor (namelyhost processor 12) from a target processor (namelyDSP16) which executes a codethis function is called "crossing" compiling and "crossing" linking.
[0045]The dynamic cross compiler 80 receives the code which a user or user agents (browser etc.) loaded by on demand one and which is not linked beforehand. A code is processed in order to analyze a non-tag code segment about the fitness

which identifies the "tag" part of (1) code or is performed by (2) DSP16.The tag part of a source code can draw to DSP the sauce in which a target is possible by predetermined markerssuch as "<start DSP code>" and "<end DSP code>" etc. which were embedded into it. When a tag part is identified in direct or analysisit is judged whether cross compillation is carried out based on the present processing state of DSP16. If it is judged that it compilesthe portion of a code will be processed by carrying out compiling of un-linking and the software which outputs a excecutable target processor code using the known compiling method. Since other tasks are performed by DSP16judgment of not compiling has the insufficient capacity (generally called available MIPS-million instruction per second) which DSP can useor when an available memory is insufficientit is madefor example. It can let the compiled code pass to the dynamic crossing linker 82and it can be immediately used in DSP16or can be saved in the DSP library 58.

[0046]The dynamic crossing linker 82 receives a non-link code beforehand. [whether it is statically stored in relation to the (1) host processor 12and] (2) It downloads dynamically to the host processor 12 via a network connection (global networksuch as the Internetis included)or is dynamically generated by the (3) dynamic cross compiler 80. The dynamic crossing linker 82 links the input code of the memory start address of DSP16 determined as run time. A memory start address can be determined from the memory map or memory table which is stored by host processor 12 or DSP16 and is managed. The dynamic crossing linker 82 changes the reference memory position in a code into the actual memory location in DSP. These memory locations can include "the branch address in a codeor refer to the data position in a code (reference)"for example.

[0047]In working examplea portable code is in COFF (common object file format) including all the information about a code including whether it is linked or not. If not linkeda symbol table defines the address which must be changed into linking a code.

[0048]There are some remarkable advantages in the above mentioned translation process compared with conventional technology. The dynamic cross compiler 80 can make a run time judgment on the 1st about where the downloaded portable code is executed. For examplein the system which has many target processors (two sets of DSP16 grades)the dynamic cross compiler 80 can compile a portable code to one arbitrary target processor based on available resources or capability. The dynamic crossing linker 82 links a code so that it may run by the target processor which does not support a relocatable code. Since a code is linked to run timethe memory location in DSP16 (or other target processors) does not need to saveand can use all the calculation resources in a device with optimum efficiency. Since compiling is attained by the knowledge of the architecture of the platform 10it can use the feature for a specific processor and platformssuch as INTEKU Gent cache architecture of the processor of one side or bothfor compiling.

[0049]ThereforeDSP16 can have various functions dynamically changed so that the throughput may be used thoroughly. For examplea user sometimes wants to load a user interface including speech recognition. the speech recognition software

which a host processor downloads software and is then performed by DSP16 -- dynamic -- cross compillation -- and a crosslink is carried out. Or based on the present position of DSP16the crosslink of the software in the DSP library 58 compiled beforehand can be carried out dynamicallyand it can be performed.
[0050]A host device driver host device driver processes the communication to the host 12 from DSP16. A driver function processes transmission of the information which took in the needed information specified in the host DSP communications protocoland passed available hard UAE Interface Division. Device drivers are RTOS dependence and communication hardware dependence.
[0051]Host DSP communications protocol (host DSP Interface Division layer)
A host DSP communications protocol governs the command between the host 12 and DSP16and communication of data. The path of some [communication]a messagea data channel and a stream**** and others It is used for a message sending initialization parameters and a command to a task. A data channel carries a lot of data in the form of a data frame between tasks and between DSP16 and the host 12. It is used for a stream letting the data made into the stream between tasks and between DSP16 and the host 12 pass.
[0052]Message each task has an entry point to the message handler which processes a message. A message is an user definition and includes the command which controls the initialization parameters and the task for a task function. A task sends a message to the host 12 via the call-back specified as the generate time. The prototype of a task message handler and the prototype of a host call-back are shown here.
[Equation 44]

[0053]A replyref parameter is a yne pull maintenance SHON dependence reference value used for returning the answer to a sending person. About each SendMessage callthe addressee has to call ReplyMessage using a replyref parameter.The actual message is as follows.
[Equation 45]

The lowest word is first sent to multi-word data.
[0054]TaskID of zero within a SendMessage function shows a system level message.It is used for a system level message carrying out a DSP-API function.
[0055]A message function is shown below.
[Equation 46]

This function sends a user definition message to the task specified by TaskID. MsgID defines a message and msgbuf contains the actual message data. Message size is countTDSPWords.An answer in a message is included in a replybuf parameterand it directs the buffer of the size replybufsize provided by the call

person. It must be sufficient size to process the answer to a specific message. An errcode parameter includes the result the following is considered to be.
[Equation 47]

[0056]
[Equation 48]

This function is used for answering a message. replyref is reference (reference) used for returning the sending person of an original message an answerand is for specific yne pull maintenance SHON. An answer is included in a buf parameter and the size is TDSPWords.An errcode parameter includes the result the following is considered to be.
[Equation 49]

[0057]It is used for a concept of a channel channel transmitting frame base data between tasks another processor or on the same processor from one processor. If generateda channel will assign a frame of a number specified that data is included and size. Firsta channel includes a list of empty frames. If a task which produces data requires an empty frame which writes in data and is written ina frame will be returned to a channel. If a task which consumes data requires a perfect frame from a channel and becomes emptya frame will be returned to a channel. By this demand and return of a frame bufferdata can be moved by the minimum copy.
[0058]Each task has an input and an output channel which were specified. Generation of a channel will direct it as an input to one taskand an output to another task. A device ID is contained in ID of a channel and the channel can let data pass by interprocessor. Channel data flow which crosses host DSP Interface Division is as follows.
[Equation 50]

A channel function is shown below.
[0059]
[Equation 51]

This function generates a data frame base communications channel. It generates the channel control structure where a count and size maintain control of 1 set of frame buffers specified in numframes and framesizerespectively. If generateda channel will assign a data frame and will add them to the empty frame list. ChannelID returns ID of a new channel. If DevID is not a thing of a call processorchannel control structure will be generated by both the call processor

and a DevID processorand the data flow which crosses a communication interface will be controlled. An errcode parameter includes the result the following is considered to be.
[Equation 52]

[0060]
[Equation 53]

This function deletes the existing channel specified by ChannelID. An errcode parameter includes the result the following is considered to be.
[Equation 54]

[0061]
[Equation 55]

This function requires an empty frame from specified low can channel ID. If Chn is NULLan output channel of a task will be used. When returningbufptr contains a pointer to a frame buffer. WaitFlag is TRUEand if there is no available frame bufferthe call person will be suspended until he can use a buffer. If WaitFlag is FALSEa function will return anyhow. An errcode parameter includes a result the following is considered to be.
[Equation 56]

[0062]
[Equation 57]

Once a task fills a frame bufferit will be returned to a channel using this function. A buffer directed by bufptr is returned to specified channel ID.If Chn is NULLan output channel of a task will be used. An errcode parameter includes a result the following is considered to be.
[Equation 58]

[0063]
[Equation 59]

This function requires all the frames of data from specified low can channel ID. If

Chn is NULLthe input channel of a task will be used. When returningthe bufptr parameter contains the pointer to the frame buffer. WaitFlag is TRUEand if there is no available perfect frame bufferthe call person will be suspended until he can use a buffer. If WaitFlag is FALSEa function will return anyhow. An errcode parameter includes the result the following is considered to be.
[Equation 60]

[0064]
[Equation 61]

The task must return a buffer to a channel using this functionif the data from a frame buffer is used. The buffer directed by bufptr is returned to specified channel ID. If Chn is NULLthe input channel of a task will be used. An errcode parameter includes the result the following is considered to be.
[Equation 62]

[0065]
[Equation 63]

This function is set as channel ID which had the input channel of a task specified. An errcode parameter includes the result the following is considered to be.
[Equation 64]

[0066]
[Equation 65]

This function is set as channel ID which had the output channel of a task specified. An errcode parameter includes the result the following is considered to be.
[Equation 66]

[0067]Although the stream stream cannot break into a frameit flows out continuously to a task and is used for the data which carries out ON. The stream consists of a circular buffer (FIFO) which accompanies the tail pointer which pursues data at the time of a head and outflow ON. Each task can have the input and output stream which were specified. The stream data flow which crosses host DSP Interface Division is as follows.
[Equation 67]

A stream function is shown below.
[0068]
[Equation 68]

This function generates a FIFO base communications stream. It generates the stream control structure of maintaining control of FIFO of the size FIFOsize. If generateda stream will assign empty FIFOand it will initialize a head and a tail pointer so that it may flow into a stream and the data flow which carries out ON may be processed. StreamID returns ID of a new stream. If DevID is not a thing of a call processorstream control structure will be generated by both a call processor and the DevID processorand the data which crosses a communication interface and flows will be controlled. An errcode parameter includes the result the following is considered to be.
[Equation 69]

[0069]
[Equation 70]

This function deletes the existing stream specified by StreamID. An errcode parameter includes the result the following is considered to be.
[Equation 71]

[0070]
[Equation 72]

This function requires the count of TDSPWords in stream FIFO specified by StrmID now.The count parameter contains the numberwhen it returns. An errcode parameter includes the result the following is considered to be.
[Equation 73]

[0071]
[Equation 74]

This function writes the count number of TDSPWords(es) in the stream specified by Strm.If Strm is NULLthe output stream of a task will be used.Data is directed

with a bufptr parameter. When it returnscounterwritten contains the TDSPWords number actually written in.An errcode parameter includes the result the following is considered to be.
[Equation 75]

[0072]
[Equation 76]

This function reads data from the stream specified by Strm. If Strm is NULLthe input stream of a task will be used. Data is stored in the buffer specified by bufptr. TDSPWords is read from a stream to maxcount.The counteread parameter includes the actual count of the read data. An errcode parameter includes the result the following is considered to be.
[Equation 77]

[0073]
[Equation 78]

This function sets the input stream of a task to specified stream ID. An errcode parameter includes the result the following is considered to be.
[Equation 79]

[0074]
[Equation 80]

This function sets the output stream of a task to specified stream ID. An errcode parameter includes the result the following is considered to be.
[Equation 81]

[0075]
[Table 3]

[0076]a system message -- the table of these defines the message which passes through between devices (namelyDSP16 from a host). Since it is used for actually carrying out routing of the message to a devicethe device ID which exists as a parameter in a corresponding function call is not built in a message. Similarlytask

ID which contains a device ID as a Johan part of a function call does not contain a device ID in a messagebut contains only the load task ID part of DSP.
[0077]
[Table 4]

[0078]
[Table 5]

[0079]
[Table 6]

[0080]Downloading native code drawing 4–drawing 6 shows working example which carries out downloading of the native code certainly and efficiently to a target processor (namelythe host 12 or DSP16). This working example that carries out downloading of the code can use a code for carrying out downloading from peripheral equipmentsuch as the Internet or other global networka locala Wide Area Network or a PC cardand a smart cardfor example.
[0081]Working example of JAVA Bean90 is shown in drawing 4and Bean90 acts as the wrapper (wrapper) of the native code 92. Bean contains some attributes 94 further written as the Code Type attribute 94athe Code Size attribute 94band the MISP demand attribute 94c. Bean90 has some actions 96 containing the Load Code action 96athe Load Parameters action 96band Exwcute Parameter96c.
[0082]In operationit is used for the Load Code action 96a loading an external native code (native for a target processor) into Bean. Since JAVA Beans has the persistence (persistence)among those Bean90 contains the native code 92 and the attribute 94it can store a part state. The Load Parameters action 96b searches a parameter from the native code 92 (using the above mentioned COFF file format for example)and stores it as attribute 94 a–c. The Execute action 96c performs the task installed in DSP16.
[0083]Signs that a code is downloaded to a target processor using Bean90 are shown in drawing 5. In this examplealthough a target processor shall be DSP16 (or one in much DSP16)it can be used also for it downloading a native code to the host processor 12. Although Bean90 of a request assumes that it resides permanently in network serverssuch as a LAN server or an internet serverBean can reside permanently in the arbitrary devices which communicate with the platforms 10such as a smart card. About the wireless data platform 10the connection with the network server 100 is often wireless.
[0084]The platform 10 is connected to the network server 100 in drawing 5. The host processor 12 shown in drawing 2 in detail can perform one or more JAVA applets 40 via the JAVA virtual machine 44. In order to download a new codethe host 12 can download Bean which loads the applet which contains Bean90 from

the network server 100or does not contain an applet from the server 100. If wrapper Bean90 is searchedthe sizethe cord type (which object for processors is a code?)and required MIPS of a native code can be asked. the host processor 12 in the architecture which shows an expected processor and drawing 5 the code 92 in having sufficient resources for an expected processor to execute the code 92or DSP16 -- it is installable so that it may come out and perform. typical -- the native code 92 -- un--- it is the code linked and compiled. Thereforethe crossing linker 82 of DSP-API50 links a code to an available memory location. Through and it install a code in the dynamic crossing linker 82and Bean executes the binary native code 92.

[0085]Typicallythat download of a native code arises is a case where the user is performing the applet which asks for a DSP function in it. Firstthe desired code is installed in DSP as the task 60or an applet confirms whether it is available in the DSP library 58. A task can be performed without download supposing it meets.

[0086]If the task is not stored in DSP16 or the DSP library 58an object (hereit is called the "DSPLoader" object) can be generatedand Bean90 can be loaded. If the DSPLoader class is local on the host 12it will be confirmed whether Bean of JAVA is also locally available. There may be Bean which has the code stored locally in the beginning. If that is rightthe code from Bean will be installed in DSP (or one which is specified by Code Type of processors). When Bean without a code is stored locallyBean can search a code from a suitable server.

[0087]On the other handif a DSPLoader object is not localJAVA loads Bean90 from the server which wrote in the applet. Nextthe code from Bean is installed as described above.

[0088]Although downloading of a native code is explained in relation to use of JAVA Beana code can also be wrapped in and attained in the language of otherssuch as an ActiveX applet.

[0089]Using JAVA Bean (or other applets) as a wrapper of a native code has a remarkable advantage. Firsta standard method simple although loading of the code is carried out to one in two or more processors may be used. Bean is generatedand a code is loaded to Bean and linked with a suitable processor. The process can take hundreds of stepswithout wrapping a code in Bean. Many pieces of a native code are combined by one appletcomplicated application is generated from the individual routine of a large number which use one appletand a routine can be combined [2nd] by request. The security function of language can be used and not only the JAVA code in Bean90 but the native code 92 is protected [3rd]. Other languagessuch as ActiveXhave a security function.

[0090]The primary importance security functions of two security are a digital signature and encryption. JAVABean or the ActiveX applet can sign with code sauceand if Bean or an applet downloadsa signature will be compared by the receiving application which has a list of reliable sauce. If there is a signature of sauce reliable to Bean or an appletit is decipherable using standard art. Thereforea native code is enciphered with the code of Bean or an applet during transmissionand unauthorized correction of a code is prevented. Since a native

code comes from the sauce which it is safe and can be trustedit is reliable with what also has an exact attribute.
[0091]Probably drawing 6 is a flow chart explaining the native code downloading process of the processor which uses JAVA Beanand a native code's being wrapped in the applet of a different language using the same art will be understanding. In Step 110digital signature Bean90 enciphered is downloaded to the device to which a JAVA virtual machine is moved. A signature is compared in Step 112. If it is not a thing from the sauce written as reliable sauceenabling (enable) of the exception handling will be carried out in Step 114. If it does not have trouble in a user with the sauce when it is what comes from the sauce which can trust Beanthe exception-handling function can give an opportunity to accept Bean in a user. If the signature is invalidthe exception handling can delete Bean90 and can send a suitable error message to a user.
[0092]If it comes from the sauce which a signature is valid and can be trustedBean will be decoded in Step 116. At this stepboth the JAVA code and the native code in Bean are decoded. In Step 118an attribute is searched from Bean90 and it is checked in Step 120 whether an applet has sufficient resources for a suitable processor to execute a code. When it does not have sufficient resourcesthe exception-handling step 114 can take the step which can refuse installation of a native code or releases resources. When there are sufficient resourcesin Step 122a code is linked using a crossing linker and installed in a desired processor. A native code is executed in Step 124.
[0093]The sample JAVA script of Bean90 is shown below.
[Table 7]

[Table 8]

[Table 9]

[Table 10]

[0094]In the above mentioned scripta NativeBean() routine generates Bean90 holding a native code. A loadCode() routine obtains a native code from a server. getFunctionalName() and a getCodeBase() routine search an attribute. An installCode() routine calls a crossing linker and loads the code which linked the native code to DSP and linked it. A loadParameters() routine orders Bean to investigate a native code and to check the attribute. getCodesize() and a getCodetype() routine are transmitted to the applet which requires an attribute.
[0095]Although the detailed explanation of this invention was turned to a certain typical working exampleif it is a person skilled in the artvarious corrections of not

only another working example but these working example will be considered. All of the correction and another working example included in Claim shall be contained in this invention.

[0096]The following paragraphs are further indicated about the above explanation.

(1) The co-processor which is a move electronic device and executes a native codeIt is a host processor system which operates so that the native code corresponding to a host processor system and a processor independent code may be executedA move electronic device containing the host processor system which operates so that the task carried out by a digital signal co-processor may be changed dynamicallysaid host processor systemand the communication circuit between said co-processors.

[0097](2) A move electronic device in which it is a move electronic device given in the 1st paragraphand said co-processor is a digital signal processor.

[0098](3) A move electronic device with which it is a move electronic device given [said] in any 1 paragraphand said processor independent code contains JAVA.

[0099](4) A move electronic device with which it is a move electronic device given [said] in any 1 paragraphand said host processor system can generate the native code of said co-processor.

[0100](5) A move electronic device which is a move electronic device given [said] in any 1 paragraphand can generate the native code of said co-processor when said host processor system compiles a processor independent source code.

[0101](6) A move electronic device with which it is a move electronic device given [said] in any 1 paragraphand said host processor system compiles the block with which the source code was identified.

[0102](7) A move electronic device which is a move electronic device given [said] in any 1 paragraphand said host processor system identifies the block of the source code which can be executed with a co-processorand compiles said block of a code.

[0103](8) A move electronic device which is a move electronic device given [said] in any 1 paragraphand contains the memory which stores in said co-processor further the library of the routine which can be downloaded and performed.

[0104](9) A move electronic device which is a move electronic device given [said] in any 1 paragraphand contains a hardware language accelerator further.

[0105](10) A move electronic device with which it is a move electronic device given [said] in any 1 paragraphand said Hardware accelerator contains a JAVA accelerator.

[0106](11) A move electronic device which is a move electronic device given in the 1st paragraphand includes the network interfacing circuit which receives data from a network further.

[0107](12) The step which is the control method of a move electronic device and executes a native code in a co-processorThe step which executes a native code and a processor independent code in a host processor systemA method containing the step which changes dynamically the task carried out by a digital signal co-

processor by said host processor systemand the step which communicates between said host processor system and said co-processor.
[0108](13) A way said step which is a method given in the 12th paragraph and executes a native code in a co-processor contains the step which executes a native code in a digital signal processor.
[0109](14) How to be a method the 12th paragraph and given in the 13th paragraphand contain the step which generates the native code of said co-processor in said common processing system further.
[0110](15) A method containing the step which said step which is a method given in the 14th paragraph and generates a native code carries out compiling of the processor independent source codeand generates a native code.
[0111](16) How to be a method given in the 15th paragraph and contain the step compiled so that the block of said source code may be identified further and it may perform with said co-processor.
[0112](17) How to be a method given in 12th paragraph-the 16th paragraphand contain the step which stores the library of the routine further downloaded and performed from said host processor system to said co-processor.
[0113](18) It is a move electronic device and they are two or more co-processors and host processor systemsOne or more sections of the source code which executes a source code and is executed with said one or more co-processors are identifiedA corresponding co-processor is determined about each section which the source code identified. About each section which the source code identifiedcompile said identified section of a code to the native code relevant to said corresponding co-processorand install in said corresponding co-processor. A move electronic device containing said host processor system which operates likesaid host processor systemand the communication circuit between said co-processors.
[0114](19) The step which is the control method of a move electronic device and executes a source code by a host processor systemThe step which identifies one or more sections of the source code executed with one or more co-processorsThe step which determines a corresponding co-processor about each section which the source code identifiedThe step which carries out compiling of said identified section of a code to the native code relevant to said corresponding co-processorand installs said native code in said corresponding co-processor about each section which the source code identifiedA method containing said host processor system and the step which performs communication between said co-processors.
[0115](20) The wireless data platform (10) containing two or more processors (1216). A communications channel is established by interprocessor so that information can be communicatedwhen a task is carried out. The dynamic cross compiler (80) carried out by one processor compiles a code to the native process code for another processor. A dynamic crossing linker (82) links the code compiled for other processors. A native code is also downloadable to a platform using JAVA Bean (90) (or other language types) which wraps it. The digital signature of JAVA

Bean can be enciphered and carried out for security.
[0116]. Apply for cross-reference this application of related application under the same dateand are incorporated here as a part of this indication. U.S. patent application 08thsuch as USURI/ No. 995600 "Mobile Communication System with Cross Compiler and Cross Linker" (owt need blanket No. 26453)U.S. patent application 08th of a brewer / No. 995597 "Method and Apparatus for ProvidingDownloadable Functionality to an EmbeddedCoprocessor." (The owt need blanket No. 26440)And it relates to U.S. patent application 08th of a brewer / No. 995603 "Method and Apparatus for Extending Security Model to Native Code" (owt need blanket No. 26439).

DESCRIPTION OF DRAWINGS

[Brief Description of the Drawings]
[Drawing 1]The block diagram of platform architecture suitable for especially general wireless data processing.
[Drawing 2]The functional block diagram of the platform of drawing 1.
[Drawing 3]The functional block diagram of dynamic crossing compiling and a dynamic crosslinking function.
[Drawing 4]Working example of the native code executed by the processor wrapped in the JAVA Bean wrapper downloaded to a device.
[Drawing 5]The figure showing operation of transmitting the native code wrapped to the processor on a device from JAVA Bean arranged on a remote server.
[Drawing 6]The flow chart which describes the security function relevant to operation of drawing 5.
[Description of Notations]
10 Wireless data platform
12 Host processors in general
14 Bus structure
16 DPS (digital signal processor)
18 Traffic controller
20 Hardware accelerator circuit
22 Video and an LCD controller
24 LCD or a video display
26 Main memory unit
28 I/O controller
30 Interface Division
32 RAM/ROM
34 Smart card
36 Keyboard
38 Mouse
40 Serial port
42 Flash memory card

44 DRAM card
46 DAC (digital/analog converter)
48 Network interface
50 DSP API (application programming interface)
52 Host DSP Interface Division layer
54 DSP device driver
56 Host RTOS (real time operating system)
58 DSP library
60 Task
62 Host DSP Interface Division layer
64 DRTOS
66 Host device driver
70 Host DSP Interface Division
80 Dynamic cross compiler
82 Dynamic crossing linker
90 JAVA Bean
92 Native code
94 Attribute
96 Action
100 Network server